BOOLEAN ALGEBRA

material prepared by: MUKESH BOHRA
MAIL UR RESPONSES@ mbthebigboss@gmail.com

Basic theorems/properties of Boolean Algebra

	Theorem/Law/Axioms	Over (+)	Over (.)
1.	Properties of 0	x+0 = x	x.0 = 0
2.	Properties of 1	x+1 = 1	x.1 = x
3.	Indempotence Law	x+x = x	x.x = x
4.	Complementarity Law	x+x' = 1	x.x' = 0
5.	Commutative Law	x+y = y+x	x.y = y.x
6.	Associative Law	x+(y+z) = (x+y)+z	x.(y.z) = (x.y).z
7.	Distributive Law	x.(y+z) = xy+xz	X+(y.z) = (x+y).(x+z)
8.	Absorption Law	x+xy = x	X(x+y) = x
9.	De-Morgan's Law	(x+y)' = x'.y'	(x.y)' = x'+y'
10.	Compliment Or Involution	(x')' = x	

Principle Of Duality:

It satates that starting with a boolean relation; another boolean relation can be derived by:

1) Changing each OR (+) sign to an AND (.) sign.
2) Changing each AND (.) to an OR (+) sign.
3) Replacing each 0 by1 & vice-versa.

e.g. The Dual of an expression x+xy = x is x.(x+y) = x

Tautology	Fallacy
If the result of a boolean expression is always TRUE or 1, it is called a tautology.	If the result of a boolean expression is always FALSE or 0, it is called fallacy.

Key Terms:

Literal	A single variable or its compliment.
Gray Code	A binary code in which each successive number differs only in one place.
Canonical form	Standard SOP or Standard POS expressions where all variables/literals are present in each term of the expression.
Maxterm	SUM term containing sum of all the literals, with or without bar
Minterm	PRODUCT term containing product of all the literals, with or without bar
K-map	It's a graphical arrangement of a truth-table in the form of a grid, which provides a simplest & systematic way of minimizing a boolean expression.

Points to remember while drawing out a K-map:

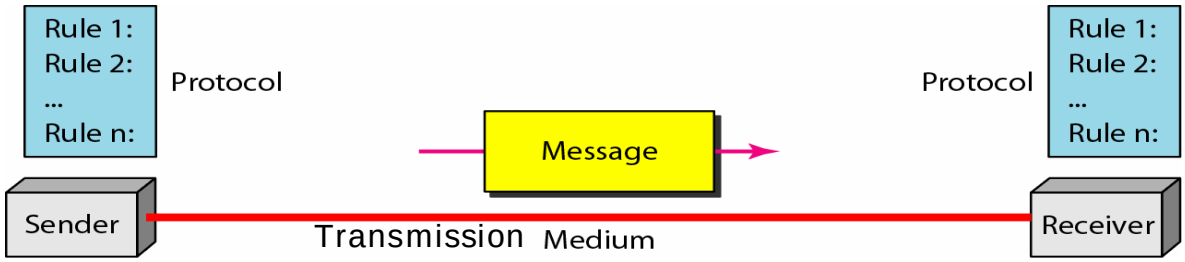
1) In SOP; each minterm is marked as binary 1 in the corresponding cell of the map.
   In POS; each maxterm is marked as binary 0 in the corresponding cell of the map.
2) While grouping the cells, check firstly for large groups. i.e. check first for a group of 16cells, then 8cells, then 4cells, then 2cells and lastly for 1cell. At each step don't forget to roll/fold the map.
3) Redundant groups should not be taken.
4) For each group, take the common row-variables and the column-variables and multiply them to get the simplified minterm (or add them to get the simplified maxterm).
   [In SOP: 0-complemented; 1-uncomplemeted]
   [In POS: 0-uncomplemented; 1-complemeted]
   Repeat the procedure for all the groups.
5) Finally, add all the minterms obtained (or multiply all the maxterms obtained) to get the final minimized expression.

Activity:

Practice the following:

1) Minimization Problems using K-maps & algebraically also.
2) Realization of a given Boolean expression using:
   - Any logic gates
   - NAND gate only
   - NOR gate only
3) To draw out the output from a logic circuit and vice-versa.
4) Writing SOP or POS forms from truth tables.
5) SOP to POS conversion and vice-versa.
6) Proving the laws or a given expression
   - Algebraically
   - Using truth–tables.

COMMUNICATIONS



SWITCHING TECHNIQUES

Circuit Switching
Complete physical connection is setup.

Packet Switching
Msg segmented into smaller packets and then transmitted into the n/w. At receving end, packets are reassembled to get back the orig msg.

Message Switching
Whole Messages are transmitted through the n/w.

TRANSMISSION MEDIA

Guided Media (Wired)
→ Includes Cables
→ Twisted-Pair Cable (@1mbps upto 100m)
Coaxial Cable (@10mbps upto several-100m)
Fiber-Optic Cables (v. high data transfer rates)

Unguided Media (Wireless)
→ Includes waves through air, water or vacuum
→ Microwaves (upto 50km)
Radio waves (city,states etc)
Infrared (v. short distance)
Satellites (across continents)

PROTOCOLS

A protocol is a **set of rules** that govern data communications. It represents an agreement between the communicating devices. Without a protocol, two devices may be connected but not communicating.

Key elements of a protocol:

- Syntax: concerns the format or structure of data blocks.
- Semantics: refers to the meaning of each section of bits.
- Timing: refers to two characteristics: when data should be sent and how fast they can be sent.

NETWORK DEVICES

MODEM	(MOdulator-DEModulator) a device that encodes data for transmission over a particular medium, such as telephone lines, coaxial cables, fiber optics, or microwaves. A modem converts digital data to analog signals and vice versa. The modem is inserted between the (digital) computer and the (analog) telephone system. Modem comes in two varieties: Internal & External.
RJ-45 CONNECTOR	Registered jack-45 is an 8-wire connector, which is commonly used to connect computers on LAN's-especially Ethernets.
ETHERNET	Ethernet is a LAN architecture developed by Xerox Corp in association with DEC and Intel. It either uses bus or star topology and support data transfer rates up to 10Mbps.
ETHERNET CARD	The computers that are a part of Ethernet, have to install a special card called Ethernet Card. It contains connections for either coaxial or twisted pair cables or both.
HUB	A hub is a network device used to connect several computers. Hubs share bandwidth among all attached devices. A hub has a number of input lines that it joins electrically. Data-frames arriving on any of the input lines are sent out on all the others. Hubs can't filter network traffic (i.e. If two packets arrive at the same time, they will collide). Hubs can be either active or passive. Hubs are only suitable for use with very lightly loaded networks.
REPEATER	A repeater is a device that amplifies the signals appearing on them.- for long distance transmission. Repeaters do not understand packets, or frames.
SWITCH	A switch is a network device that is used to segment networks into smaller subnets or LAN segments. Segmenting the n/w into small subnets, prevent traffic overloading.
BRIDGE	A bridge is a network device that connects two or more LANs. When a data-frame arrives, software in the bridge extracts the destination address (MAC) and looks it up in a table to see where to send the frame. Bridges can filter network traffic.
ROUTER	A Router is a network component that joins several networks together intelligently. It works like a bridge, but it can handle different protocols. A router is more powerful than a bridge because it can look up the best route to a distant site. The router filters network traffic based on IP addresses. The Internet relies heavily on routers.
GATEWAY	A gateway is network device that connects dissimilar networks. It establishes an intelligent connection b/w a local network and external networks with completely different structures.

Good Network Design: The 80-20 Rule

[80% of the traffic on a given network segment should be local]

TIP

- ✓ Place the SERVER at a place (or building) where the total number of computers connected is maximum and the sum of distances for others places (or buildings) is minimum.

ABBREVIATIONS

Abbreviation	Expanded Form
ARPANET	Advanced Research Projects Agency NETwork
NSFNET	National Science Foundation NETwork
NIU	Network Interface Unit
NIC	Network Interface Card
MAC	Medium Access Control
bps	bits per second
BPS	Byte Per Second
LAN	Local Area Network
MAN	Metropolitan Area Network
WAN	Wide Area Network
MODEM	Modulator/Demodulator
AM	Amplitude Modulation
FM	Frequency Modulation
PM	Phase Modulation
RJ-45	Registered Jack-45
PSTN	Public Switched Telephone Network
PSDN	Public Switched Data Network
ISDN	Integrated Services Digital Network
HTTP	Hypertext Transfer Protocol
FTP	File Transfer Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
SLIP	Serial Line Internet Protocol
PPP	Point To Point Protocol
POP	Post Office Protocol
IMAP	Internet Mail Access Protocol
SMTP	Simple Mail Transfer Protocol
MIME	Multipurpose Internet Mail Extensions
URL	Uniform Resource Locator
DNS	Domain Name Server

GSM	Global Systems for Mobile communications
TDMA	Time Division Multiple Access
CDMA	Code Division Multiple Access
SIM	Time Division Multiple Access
GPRS	General Packet Radio Service
WLL	Wireless in Local Loop
WAP	Wireless Application Protocol
3G	3rd Generation for mobile communications
EDGE	Enhanced Data rates for Global Evolution
SMS	Short Message Service
MMS	Multimedia Message Service

e-mail	electronic mail

www	world wide web

HTML	HyperText Markup Language
DHTML	Dynamic HyperText Markup Language
XML	eXtensible Markup Language

JSP	Java Server Pages
PHP	Preprocessor Hypertext
ASP	Active Sever Pages

OSS	Open Source Software
FLOSS	Free Libre and Open Source Software
GNU	GNU's Not Unix
FSF	Free Software Foundation
OSI	Open Source Initiative
W3C	World Wide Web Consortium

KEY TERMS

Internet: The Internet is a world-wide computer network, i.e., a network that interconnects millions of computing devices throughout the world.

Intranet: An internet used by a single organization for internal purposes along with the key internet applications, especially the WWW. e.g. banks use intranet.

Interspace: Interspace allows multiple users to communicate online with real-time audio, video and text chat in 3D environments.

Telnet: Telnet is an Internet utility that lets you logon to a remote computer and function as if directly connected to that computer.

FTP (File Transfer Protocol): is used to send files from one system to another under user command. Both text & binary files are accommodated, and the protocol provides features for controlling user access.

HTTP (Hyper Text Transfer Protocol): is the foundation protocol of the World Wide Web (WWW) and can be used in any client/server application involving hypertext. The most typical use of HTTP is b/w a web-browser and a web-server. When a browser wants a Web page, it sends the name of the page it wants to the server using HTTP; the server then sends the page back.

Network Security Concepts:

Authorization	Means permissions or granting access to a service.
Authentication	Concerned with assuring that a communication is authentic. It is the process of verifying the identity claimed by a communicating entity.
Firewall	The system designed to prevent unauthorized access to or from a private network is called firewall. A firewall is considered as a first line of defense in protecting private information. All traffic from inside to outside, and vice versa, is made to pass through the firewall. The firewall forms a barrier which acts as a filter and only authorized traffic as defined by the local security policy, will be allowed to pass. A firewall can be implemented in both hardware or software or both.
Cookies	Cookies are messages that web sites use to recognize users who have previously visited them. The browser stores the message in a text file (with a few parameters like name, value, expiration date etc.) The next time the user accesses that site, the information in the cookie is sent back to the site and the customized web page is opened.
Hackers	An exceptionally enthusiastic and skilled person who “breaks into” computers without authorization;
Crackers	The crackers are the malicious programmers who break the security of a computer system, software program, algorithm, encrypted data, and so on. (u might heard of password cracking, software cracking)
Hacking	Hacking refers to the unauthorized access of information.
Cyberlaw	Cyberlaw is a generic term, which refers to all the legal and regulatory aspects of internet and the World Wide Web. Applications: track activities on internet, handling issues related to digital transactions, etc.
Virus	A Malicious Program that attaches itself to a program/file and propagates copies of itself to other programs IN order to infect them.
Trojan Horse	A Trojan Horse is useful, or apparently useful, computer program containing a hidden code that, when invoked, performs some unwanted or harmful function (such as erasing the hard-disk on a specified date).
Worms	A worm is a computer program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again. Worms are found primarily on computers that are capable of multitasking and are connected by a network.
Spam	Spam refers to electronic junk mail or junk newsgroup postings.

Comparison between LAN, WAN & MAN

Parameter	LAN	WAN	MAN
Area covered	Covers small area i.e. within the building	Covers large geographical area (across countries, continents)	Covers larger than LAN & smaller than WAN (within cities, town)
Error rates	Lowest	Highest	Moderate
Transmission speed	High speed	Low speed	Moderate speed
Equipment cost	Uses inexpensive equipment	Uses most expensive equipment	Uses moderately expensive equipment.

NETWORK TOPOLOGY: means the way systems are connected in a network.

TOPOLOGY	ADVANTAGES	DISADVANTAGES
BUS	Short cable length, easy to extend.	Fault diagnosis difficult, nodes must be intelligent.
RING	Short cable length, no wiring closet space required.	Fault diagnosis difficult, single node failure causes network failure.
STAR	Centralized control, fault diagnosis and isolation easier, simple access protocols.	Long cable length, difficult to expand, central node dependency.
TREE	Hierarchical flow of data, easy to extend.	Long cable length, root dependency.

Terms related to WWW

Web Server	A web server refers to a location on the internet that contains information in the form of web pages.
Web Page	A web page refers to a document on the web.
Website	A web site comprises of a collection of web pages on a net server that may be maintained & updated by an organization or individuals.
Web Portal	A website that hosts other websites.
URL	Each website has a unique web address called URL (Uniform Resource Locator).
Domain	A portion of the Internet distinguished by a particular final part of the name. For instance, www.google.com, is a server in the commercial (.com) domain. Some most common domains are: com, edu, gov, org, net, co, mil, etc. In addition, some domain names are location based also; which includes two letter abbreviations for country names, like .in for India, .au for Australia, .uk for United Kingdom, .jp for Japan. e.g. www.cbse.nic.in
Web Browser	A software program that is used to view web pages. Browser helps you to connect to the web sites. Ex. Internet Explorer, Mozilla Firefox, Opera, Google Chrome, etc.
Web Hosting	Means hosting web server application on a computer system.
Web Scripting	Process of creating and embedding scripts in a web page.
Script	A script is a list of commands embedded in a web page.

DATA STRUCTURES

The logical or mathematical model of data is called a data structure. Data structures are the building blocks of a program. The selection of a particular data structure will help the programmer to design more efficient programs.
A data structure has a well defined operations, behaviour & properties.

Operations on a Data Structure:

Insertion	Deletion	Searching	Sorting	Traversal
Reversing	Merging	Copying	Concatenation	

Types of Data Structures

Linear Data structures
↓
elements form a sequence
↓
- Array
- Linked List
- Stack (LIFO List)
- Queue (FIFO List)

Non Linear Data Structures
↓
represent hierarchical relationships
↓
- Trees
- Graphs

ARRAYS

An array is a finite collection of similar elements stored in contiguous memory locations.

Array size = U- L + 1 In C++, the lower bound (L) is always 0.

The address of first element of an array is called the Base Address (B).

2-D arrays: A 2-D array is an array in which each element is itself an array. It's sometimes also called a matrix.

Formula for address calculation in 2-D arrays

In Row Major Arrangement (row wise)	In Column Major Arrangement (column wise)
Address of a[i][j] = B + W (i *col + j) Or Address of a[i][j] = B + W [(i –Lr)* col + (j-Lc)]	Address of a[i][j] = B + W (j *row + i) Or Address of a[i][j] = B + W [(j –Lc)* row + (i -Lr)]

Here, Lr = first row number and Lc = first column number

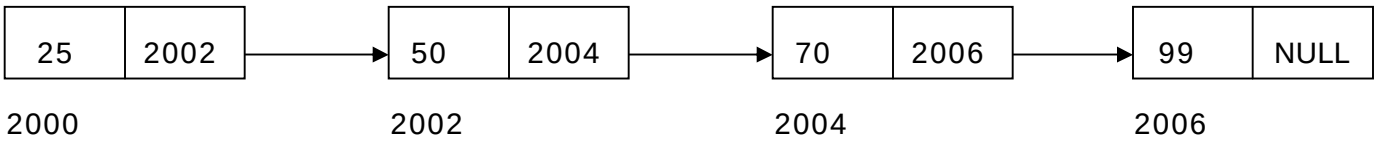
SEARCHING AND SORTING:

(Once Go Through the Searching and Sorting Algorithms)

SEARCHING	SORTING
LINEARS SEARCH BINARY SEARCH	INSERTION SORT SELECTION SORT BUBBLE SORT

LINKED LIST

A linked list is a linear collection of specially designed elements called nodes; each of which stores two items of information- an element of the list (data part) and a link (pointer).



A linked list can grow as well as shrink in size during its lifetime. Linked lists are used preferably when the quantity of data is not known prior to execution.

Defining each node of a linked list: In linked lists, data is stored in the form of nodes and at run-time, memory is allocated for creating nodes (using new operator). The data can be accessed using the START pointer of the list.

```
struct node
{
   int data;            // data part
   node *link;          // link part
};
```

STACK

A stack is a linear data structure in which addition of new element or deletion of an existing element takes place at the same end. This end is often known as the top of the stack.

The stack is sometimes also called as LIFO List (Last-In-First-Out), because the last element pushed into the stack is the first one to be popped out.

Operations on a stack

Operation	DESCRIPTION
PUSH	To insert an element at the top of the stack
POP	To remove an existing element from the top of the stack

STACK IMPLEMENTATION

Static implementation (using arrays)

Dynamic implementation (using pointers)

PUSH and POP function definitions (using arrays)

```
void stack :: push ( int item )
{
        if ( top == MAX - 1 )
        {
                cout << "Stack is full" ;
                return ;
        }
        top= top+ 1 ;
        arr[top] = item ;
}
```

```
void stack :: pop( )
{
        if ( top == -1 )
        {
                cout << "Stack is empty" ;
                return NULL ;
        }

        int d = arr[top] ;
        top = top - 1 ;
        cout<<“Popped item is:”<<d;


}
```

PUSH and POP function definitions (using pointers)

```
void stack :: push ( int item )
{
        node *temp ;
        temp = new node ;          // dynamic allocation


        temp -> data = item ;
        temp -> link = top ;
        top = temp ;               //top pointer is set
}
```

```
void stack :: pop( )
{
        if ( top == NULL )
        {
              cout<< "Stack is empty" ;
              return NULL ;
        }
        node *temp ;
        temp = top ;
        int d = temp -> data;
        cout<<“Popped item is:”<<d;

        top = top -> link ;        //resetting the top pointer
        delete temp ;              //freeing the popped node
}
```

QUEUE

Queue is linear data structure that permits insertion of new element at one end (called rear of the queue) and deletion of an existing element at the other end (called front of the queue). Queues are sometimes also called FIFO List (First-In-First-Out).

Operations on a stack →

Operation	DESCRIPTION
Insertion	To add a new element at the rear
Deletion	To remove an element from the front

QUEUE IMPLEMENTATION

Static implementation (using arrays)

Dynamic implementation (using pointers)

INSERT and DELETE function definitions (using arrays)

```
void queue :: insertq ( int item )
{
        if ( rear == MAX - 1 )
        {
                cout << "Queue is full" ;
                return ;
        }

        Rear = rear + 1 ;
        arr[rear] = item ;

        if ( front == -1 )
                front = 0 ;

}
```

```
void queue :: delq( )
{
        if ( front == -1 )
        {
                cout << "Queue is Empty" ;
                return NULL ;
        }

        int d = arr[front] ;
        arr[front] = 0 ;
        if ( front == rear )
                front = rear = -1 ;
        else
                front = front + 1; ;

        cout<<“deleted item is:”<<d;
}
```

INSERT and DELETE function definitions (using pointers)

```
void queue :: insertq ( int item )
{
        node *temp ;
        temp = new node ;          // dynamic allocation

        temp -> data = item ;
        temp -> link = NULL ;

        if ( front == NULL )
        {
                rear = front = temp ;
                return ;
        }

        rear -> link = temp ;
        rear = rear -> link ;
}
```

```
void queue :: delq( )
{
        if ( front == NULL )
        {
                cout << "Queue is empty" ;
                return NULL ;
        }

        node *temp ;
        int d = front  -> data ;
        cout<<“deleted item is:”<<d;

        temp = front ;
        front = front -> link ;
        delete temp ;          //freeing the deleted node

}
```

Function That Evaluates The Postfix Expression

```
void postfix :: calculate( )
{
        int n1, n2, n3 ;
        while ( *s )
        {
                if ( *s == ' ' || *s == '\t' )                  // skip whitespace, if any
                {
                        s++ ;
                        continue ;
                }

                if ( isdigit ( *s ) )                           // if digit is encountered
                {
                        nn = *s - '0' ;
                        push ( nn ) ;
                }
                else
                {                                               // if operator is encountered
                        n1 = pop( ) ;
                        n2 = pop( ) ;
                        switch ( *s )
                        {
                                case '+' :
                                        n3 = n2 + n1 ;
                                        break ;
                                case '-' :
                                        n3 = n2 - n1 ;
                                        break ;
                                case '/' :
                                        n3 = n2 / n1 ;
                                        break ;
                                case '*' :
                                        n3 = n2 * n1 ;
                                        break ;
                                case '%' :
                                        n3 = n2 % n1 ;
                                        break ;
                                case '$' :
                                        n3 = pow ( n2 , n1 ) ;
                                        break ;
                                default :
                                        cout << "Unknown operator" ;
                                        exit ( 1 ) ;
                        }
                        push ( n3 ) ;
                }
                s++ ;
        }
}
```

DATA FILE HANDLING

In C++, a file, at its lowest level, is interpreted simply as a sequence of or *stream* of bytes.

Files in computers are of two types. One that stores instruction for the computer, i.e. a program file and the second that stores data, i.e. data file.

DATA FILES

Text Files

data stored in the form of characters (ASCII Code)

each line terminated with a special Character known as end-of-line (EOL) character

Binary Files

data stored in the form of sequence of bytes, i.e. 0's & 1's

there is no EOL character

What is a Stream?

A stream is a general name given to the flow of data. Different streams are used to represent different kind of data flow. Each stream is associated with a particular class of ***fstream.h*** header file of the standard library.

The streams are of the following types:

Type of stream	Associated Class
input stream (Read mode)	ifstream
output stream (Write mode)	ofstream
input/output stream (R/W mode)	fstream

Basic Operations On A Text File

- Creating or writing in file.
- Reading a text file and displaying contents.
- Manipulating the contents read from a text file.

To be able to carry out the above basic operations on a file, the following sequence has to be followed:

- ✓ Open the file.
- ✓ Perform operation on the file.
- ✓ Close the file.

Opening A File

Before opening a file, we shall create a file stream object of a particular class (ifstream, ofstream or fstream) depending upon the type of operation.
e.g. in order to open a file as an input file i.e. data will be read from it and no other operation would take place,we shall create a file stream object of ifstream type.
Similarly, in order to open an output file (on which no operation can take place except writing only), we shall create a file stream object of ofstream type.

A file can be opened in two ways:

- ✧ Using the constructor function of the class (useful when we use only one file in the stream)
  e.g. ofstream outfile("marks.dat");

- ✧ Using member function open() of the class (useful in case of multiple files)
  e.g. ofstream outfile;
  outfile.open("marks.dat");

Concept Of File Modes

The file mode describes how a file is to be used : to read from it, to write to it, to append it, and so on.

File Mode Constants	Meaning	Associated Class
ios::in	Open for reading (default for ifstream)	ifstream
ios::out	Open for writing (default for ofstream)	ofstream
ios::ate	Start reading or writing at the end of file (AT End)	ofstream ifstream
ios::app	Start writing at end of file (APPend)	ofstream
ios::trunk	Truncate file to zero length if it exists (TRUNCate)	ofstream
ios::nocreate	Error when opening if file doesnot already exist.	ofstream
ios::noreplace	Error when opening for output if file already exist, unless ate or app is set.	ofstream
ios::binary	Open file in binary (not text) mode.	ofstream ifstream

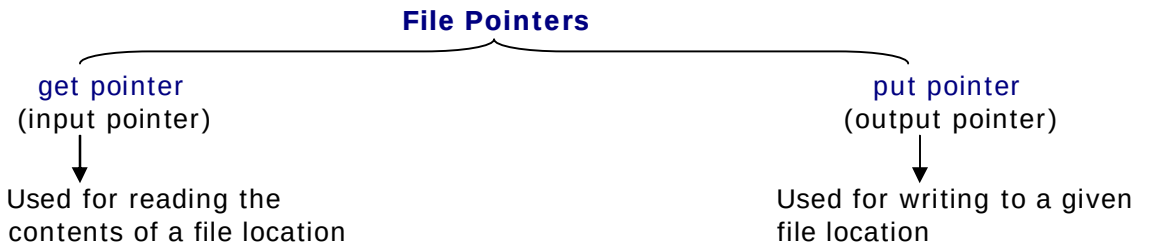
Note: 1) we can combine two or more file mode constants using the C++ bitwise OR operator.
2) The fstream class does not a mode by default and, therefore, one must specify the mode explicitly when using an object of the fstream class.

How is end-of-file detected in a file?

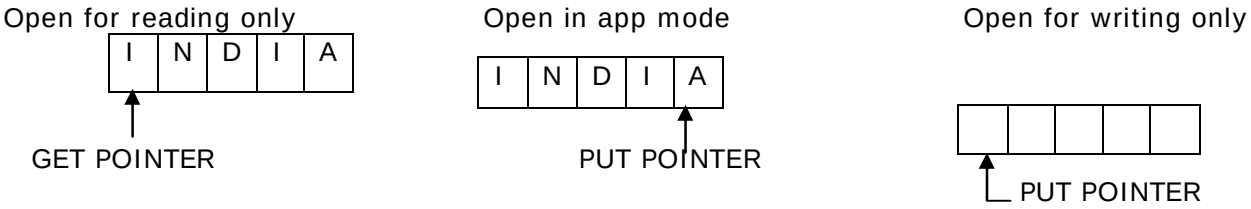
The end of any file is checked with eof() function which is predefined in class ios of ifstream, ofstream and fstream classes. If the file pointer points to the end-of-file then condition is TRUE and the object returns zero otherwise it returns a non-zero value.

File Pointers

Each file has two pointers associated with it which are called file pointers. One of them is the input pointer, known as *get pointer*; and the other one is called output pointer or the *put pointer*.



NOTE: Each time an input or output operation takes place, the concerned pointer is automatically advanced.



How can the file pointers be moved in a file?

The file stream classes support some predefined functions that navigate the position of the file-pointers. The relevant functions are: seekg(), seekp(), tellg(), and tellp().

Function name	Description
Seekg()	Moves the get pointer or input pointer to a specified location
seekp()	Moves the put pointer or the output pointer to a specified location
tellg()	Gives the current position of the get pointer
tellp()	Gives the current position of the put pointer

seekg() and seekp() are the functions used for manipulation of file pointers.
e.g. ifile.seekg(30);
This statement moves the get pointer to the byte number 30 in the file linked with ifile. Remember the counting of bytes in a file begins from zero.

⇒ *Another form of seek() function*

```
Seekg( offset, reposition);
Seekp( offset, reposition);
```

```
e.g.   ifile.seekg(10, ios::beg);    //goto byte number 10 from the begining
       ifile.seekg(10, ios::cur);    //goto byte number 10 from the current position
       ifile.seekg(0, ios::end);     //goto end of the file.
       Ifile.seekg(-10, ios::end)  // goto 10 bytes before the end of file.
```

Closing A File

A file is closed by disconnecting it with the stream it is associated with. A file can be CLOSed in two ways:

- ✧ If the file has been opened using constructor, it gets closed automatically as soon as the stream objects go out of scope. This calls the destructor, which closes the file.
- ✧ Using close function of the class
  e.g. infile.close();
  (or) outfile.close();

EXAMPLE 1: CREATING/WRITING A FILE:

```
#include<fstream.h>
Void main()
{  ofstream outfile("poem.txt");     // create file for output
   outfile<<"I love my country";     // send text to file
   outfile<<"\n I love my India";
}
```

NOTE:

1) In the above program, getfrom operator << is appropriately overloaded; we used it to write text to the file.
2) When the program terminates, the outfile object goes out of scope. This calls the destructor, which closes the file. i.e. we don't need to close the file explicitly.
3) When the program is executed, the lines of text specified in the program are written into the file. There is no output to the screen. To see what the text is in the file poem.txt, goto Dos shell & type the following: type poem.txt and press enter; it will show the file contents.

TIP: poem.txt is the physical name of the file & outfile is the logical name of the file.

EXAMPLE 2: READING A FILE: we can read the above file at a later stage as (first we must create an object of class ifstream):

```
#include<fstream.h>
Void main()
{   const int max=80;                 // size of buffer
    char line_read[max];              //character buffer
    ifstream infile("poem.txt");
    while(infile)                              // until end of file
    {   infile.getline(line_read, max);        //read a line until \n is encountered
        Cout<<buffer;                 // display it
     }
}
```

OUTPUT: I love my country
I love my India

Problems

QUES 1) Write a function in C++ to count the number of lowercase alphabets present in a text file “BOOK.TXT”.

SOL)

```
void LowerLetters( )
{ clrscr( );
ifstream fin("BOOK.TXT",ios::in);
char ch;
int lowercount=0;
while(fin)
{fin.get(ch);
if(islower(ch))
lowercount++;
}
cout<<"\nTotal number of Lowercase alphabets in the file = "<<lowercount;
getch( );
}
```

QUES 2) Given a binary file PHONE.DAT, containing records of the following structure type

```
class phonlist
{ char Name[20] ;
char Address[30] ;
char AreaCode[5] ;
char PhoneNo[15] ;
public ;
void Register( ) ;
void Show( ) ;
int CheckCode(char AC[ ])
{ return strcmp(AreaCode, AC) ;
}
} ;
```

Write a function TRANSFER( ) in C++, that would copy all those records which are having AreaCode as “DEL” from PHONE.DAT to PHONBACK.DAT.

SOL)

```
void TRANSFER( )
{ ifstream fin(“PHONE.DAT’,ios::in,ios::binary);
ofstream fout(“PHONEBACK.DAT”,ios::out,ios::binary);
phonlist P;
while(fin) // or while(!fin.eof( ))
{ fin.read((char*)&P,sizeof(P));
if(P.CheckCode(“DEL”)= = 0)
fout.write((char*)&P,sizeof(P));
}
fin.close( );
fout.close( );
}
```

QUES 3) Given a binary file GAME.DAT, containing records of the following structure type

```
struct Game
{ char GameName[20] ;
char Participate[10][30] ;
} ;
```

Write a function in C++ that would read contents from the file GAME.DAT and creates a file named BASKET.DAT copying only those records from GAME.DAT where the game name is “Basket Ball”.

SOL)

```
void BPlayers( )
{ ifstream fin(“GAME.DAT’,ios::in,ios::binary););
ofstream fout(“BASKET.DAT”,ios::out|ios::binary);
```

```
Game G;
while(fin) // or while(!fin.eof( ))
{ fin.read((char*)&G,sizeof(Game));
   if(strcmp(G.GameName,"Basket Ball")= = 0)
fout.write((char*)&G,sizeof(G));
}
fin.close( );
fout.close( );
}
```

QUES 4) Given a binary file SPORTS.DAT,containg records of the following structure type :

```
struct Sports
{ char Event[20] ;
char Participant[10][30] ;
} ;
```

Write a function in C++ that would read contents from the file SPORTS.DAT and creates a file named ATHLETIC.DAT copying only those records from SPORTS.DAT where the event name is "Athletics".

SOL)

```
void AthletsList( )
{ ifstream fin("SPORTS.DAT',ios::in,ios::binary););
ofstream fout("ATHLETIC.DAT",ios::out|ios::binary);
Sports S;
while(fin) // or while(!fin.eof( ))
{ fin.read((char*)&S,sizeof(Sports));
if(strcmp(S.Event,"Athletics")= = 0)
fout.write((char*)&S,sizeof(S));
}
fin.close( );
fout.close( );
}
```

QUES 5)

```
void main( )
{ char ch = 'A' ;
fstream fileout("data.dat", ios::out) ;
fileout<<ch ;
int p = fileout.tellg( )
cout<<p ;
}
```

What is the output if the file content before the execution of the program is the string "ABC". (Note that " " are not part of the file).

SOL) 1 (Since, the file is opened in out mode, it looses all the previous content, if the file mode is app, then result will be 4).

QUES 6) Write a function to count the number of words present in a text file named "PARA.TXT". Assume that each word is separated by a single blank/space character and no blanks/spaces in the beginning and end of the file.

SOL)

```
void WordsCount( )
{ clrscr( );
ifstream fin("PARA.TXT",ios::in);
char ch;
int Words=1;
if(!fin)
{ cout<<"No words at all in the file";
exit(0);
}
while(fin)
{  fin.get(ch);
   if(ch= =' ')
   words++;  }
```

```
cout<<"\nTotal number of Words in the file = "<<Words;
getch( );
}
```

QUES 7) Following is the structure of each record in a data file named “COLONY.DAT”

```
struct COLONY
{ char Colony_Code[10] ;
char Colony_Name[10]
int No_of_People ;
} ;
```

Write a function in C++ to update the file with a new value of No_of_People. The value of Colony_Code and No_of_People are read during the execution of the program.

SOL)

```
void Update( )
{ fstream finout(“COLONY.DAT”,ios::in|ios::out);
COLONY C;
finout.seekg(0);
while(finout)
{ finout.read((char *)&C, sizeof(C));
cout<<”\nThe Colony Code is “<<C.Colony_Code;
cout<<”\nThe Colony Name is”<<C.Colony_Name;
cout<<”\nEnter the Number of People”;
cin>>C.No_of_People;
finout.seekp(finout.seekp( )-sizeof(C));
finout.write((char *)&C,sizeof(C));
}
}
```

QUES 8)

```
void main( )
{ char ch = ‘A’ ;
fstream fileout(“data.dat”, ios :: app) ;
fileout<<ch ;
int p = fileout.tellg( ) ;
cout << p ;
}
```

What is the output if the file content before the execution of the program is the string ? “ABC”
(Note that “ “ are not part of the file)

SOL) 4 (Since, the file is opened in app mode, it retains the previous content also, if the file mode is out, then result will be 0 since it will loose all the old content of the file.)

QUES 9) Write a function to count the number of blanks present in a text file named “PARA.TXT” .

SOL)

```
void BlanksCount( )
{ clrscr( );
ifstream fin("PARA.TXT",ios::in);
char ch;
int Blanks=0;
if(!fin)
{ cout<<”No words at all in the file. So no blank spaces”;
exit(0);
}
while(fin)
{  fin.get(ch);
   if(ch= =’ ‘)
   Blanks++;   }
cout<<"\nTotal number of Blank Spaces in the file = "<<Blanks;
getch( );
}
```

QUES 10) Following is the structure of each record in a data file named "PRODUCT.DAT"

```
struct PRODUCT
{ char Product_Code[10] ;
char Product_Description[10] ;
int Stock ;   } ;
```

Write a function in C++ to update the file with a new value of Stock. The Stock and the Product_Code, whose Stock to be updated, are read during the execution of the program.

SOL)

```
void Update( )
{ fstream finout("PRODUCT.DAT",ios::in|ios::out);
PRODUCT P;
finout.seekg(0);
while(finout)
{ finout.read((char *)&P, sizeof(P));
cout<<"\nThe Product Code is "<<P.Product_Code;
cout<<"\nThe Product Description is "<<P.Product_Description;
cout<<"\nEnter the Stock: ";
cin>>P.Stock;
finout.seekp(finout.seekp( )-sizeof(P));
finout.write((char *)&P,sizeof(P));
}
}
```

QUES 11) Given a binary file TELEPHON.DAT, containing records of the following class Directory :

```
class Directory
{ char Name[20] ;
char Address[30] ;
char AreaCode[5] ;
char phone_No[15] ;
public ;
void Register( ) ;
void Show( ) ;
int CheckCode(char AC[ ])
{ return strcmp(AreaCode, AC) ;
}
} ;
```

Write a function COPYABC( ) in C++, that would copy all those records having AreaCode as "123" from TELEPHON.DAT to TELEBACK.DAT.

SOL)

```
void COPYABC( )
{ ifstream fin("TELEPHON.DAT',ios::in|ios::binary);
ofstream fout("TELEBACK.DAT",ios::out,ios|binary);
Directory D;
while(fin) // or while(!fin.eof( ))
{ fin.read((char*)&D,sizeof(D));
if(D.CheckCode("123")= = 0)
fout.write((char*)&D,sizeof(D));
}
fin.close( );
fout.close( );
}
```

QUES 12) Assuming the class FLOPPYBOX, write a function in C++ to perform following:

(i) Write the objects of FLOPPYBOX to a binary file.

(ii) Reads the objects of FLOPPYBOX from binary file and display them on screen.

```
class FLOPPYBOX
{ int size;
char name[10];
public:
void getdata(){cin>>size;gets(name);}
void showdata(){cout<<size<<" "<<name<<endl;}
};
```

QUES 13) Write a function in C++ to count and display the number of lines not starting with alphabet 'A' present in a text file "STORY.TXT".

Example:

If the file "STORY.TXT" contains the following lines,

The rose is red.

A girl is playing there.

There is a playground.

An aeroplane is in the sky.

Numbers are not allowed in the password.

The function should display the output as 3

QUES 14) Write a function in C++ to count the number of uppercase alphabets present in a text file "ARTICLE.TXT".

SOL)

```
void UpperLetters( )
{ clrscr( );
ifstream fin("ARTICLE.TXT",ios::in);
char ch;
int uppercount=0;
while(fin)
{ fin.get(ch);
if(isupper(ch))
uppercount++;  }
cout<<"\nTotal number of Uppercase alphabets in the file = "<<uppercount;
getch( );
}
```

QUES 15) Write a program to count the number of wordsin a text file

SOL)

```
void main()
{ char line[80];
ofstream ofile("data.txt");
cout<<" \n enter a string to be stored in a file";
gets(line);
for(int i=0;i<strlen(line);i++)
   ofile.put(line[i]);
ofile.close();
int word=0;
ifstream("data.txt");
ifile.getline(line, 80);
for(i=0;i<<strlen(line);i++)
   if(line[i]==' ' || line[i]=='.')
      word++;
cout<<line;
cout<<"\n no of words in string are"<<word;
getch(); }
```
